CS 4414 Machine Problem 5 (MP5)

Due December 4th, 2014

Purpose

The primary objective of this homework is to familiarize you with remote procedure calls (RPC) and remote file systems. As secondary objectives, you will become familiar with the well-known file transfer protocol, FTP, and with Berkeley sockets (also known as the BSD socket API).

Problem

You are to implement a very simple, bare-bones FTP server that will allow off-the-shelf FTP clients to list, store, and retrieve files from the local file system.

FTP exemplifies client-server architecture, and you will be creating a simple server implementation. The protocol is enacted through a pair of TCP connections: (1) a control connection that is accepted by the Server in which the Client provides commands and the Server sends response codes; and (2) separate data connections that are (nominally) accepted by the Client in which data is exchanged (e.g., files are uploaded, downloaded, and directory listings are sent).

Your Server will have one command-line parameter: the port number upon which to listen and accept incoming control connections from Clients. (Typically you'll need to use a port number greater than 1024, and even then, you'll have to check the return code for listen() to make sure that you're successfully listening on that port.)

Your Server will implement the bare-bones requirements/commands set forth in Section 5.1 of the FTP specification, IETF RFC 959, with the following exceptions:

You do not need to support RECORD structure (only FILE). If the Client attempts to use the "STRU R" command to change from FILE (the default) mode, your Server should respond with "*504 Command not implemented for that parameter*".

We are only interested in storing/retrieving binary copies of files, i.e., byte-by-byte copies, where a byte is 8-bits. This is the *Image* type. You do not need to support the default file type, *ASCII Non-print*, which would entail decoding the data back to an internal character representation from the Client- encoded 8-bit NVT-ASCII representation (way too much work). This means that you must support the "TYPE I" command that indicates the Client wishes binary image mode. (Command-line ftp clients typically issue this command to the server when the user types "binary" at the prompt.) If a client attempts a STOR or RETR command without having first switched to image mode, you should issue a "*451 Requested action aborted: local error in processing*" response.

You need to implement the LIST command.

Other details:

- Your Server need only support one active client at a time for this assignment.
- Your Server should, at minimum, interoperate with the default ftp client found on the power nodes (/usr/bin/ftp)
- The "root" of the file namespace serviced by your Server program will be the current

working (local) directory when it was executed.

Resources you should use:

- The official FTP specification, IETF RFC 959 (http://www.ietf.org/rfc/rfc0959.txt)
- You will probably find Sections 4, 5, and 6 most useful.
- Fluffy description of FTP protocol (http://en.wikipedia.org/wiki/File_Transfer_Protocol)
- Convenient list of all raw FTP commands (http://www.nsftools.com/tips/RawFTP.htm)
- Verbose examples of FTP sessions:
- Section 7 of RFC 959
- http://www.cs.colostate.edu/helpdocs/ftp.html
- Examples for using BSD sockets, including code for establishing a server-socket to listen upon (e.g., for the incoming Client control connection), and for connecting to a remote server-socket (e.g., setting up a data connection back to the Client in response to a PORT command):http://en.wikipedia.org/wiki/Berkeley_sockets

Miscellaneous

- You must use Unix/Linux for this homework.
- Your implementation must be in C/C++ and use the POSIX/BSD APIs for manipulating file data and Internet sockets. You must use the GNU C/C++ development tools (e.g., g++).
- You must submit your solution by uploading an archive (e.g., .tar, .tgz, .zip) of your source file(s) to the course's Collab portal. Along with your sources (.c/.cpp/.h files), you must submit a Makefile that will allow us to compile your solution, specifically to an executable named "my_ftpd"
- You must submit in class a printed writeup regarding your solution in accordance with the requirements set in the Beginning of Course Memo (BOCM). The writeup must have your source files appended.

If you have any other questions about the homework please ask.